Enhancing the Completeness and Accuracy of Knowledge-Based Activity Models through Incremental Data-Driven Learning

Gorka Azkune^a, Aitor Almeida^a, Diego López-de-Ipiña^a, Liming Chen^b

^a *DeustoTech - University of Deusto, Avenida de las Universidades 24, 48007 Bilbao (Spain)*

^b *School of Computer Science and Informatics - De Montfort University, LE19BH Leicester (United Kingdom)*

Abstract

Knowledge-driven activity recognition is an emerging and promising research area which has already shown very interesting features and advantages. However, there are also some drawbacks, such as the usage of generic and static activity models. This paper presents an approach to using data-driven techniques to evolve knowledge-driven activity models with a user's behavioral data. The approach includes a novel clustering process where initial incomplete models developed through knowledge engineering are used to detect activity clusters and aggregate new actions. Based on those activity clusters, a learning process is then designed to learn and model varying ways of performing activities in order to acquire complete and specialized activity models. The approach has been tested with real users' inputs, noisy sensors and demanding activity sequences. Initial results have shown that complete and specialized activity models are properly learned with success rates of 100% at the expense of learning some false positive models.

Keywords: Activity Recognition, Knowledge-driven, Learning, Activity Model

Email addresses: gorka.azkune@deusto.es (Gorka Azkune), aitor.almeida@deusto.es (Aitor Almeida), dipina@deusto.es (Diego López-de-Ipiña), liming.chen@dmu.ac.uk (Liming Chen)

1. Introduction

Human activity recognition has become an important research topic in areas such as ambient assisted living [1], social robotics [2], surveillance-based security [3] and context-aware computing [4]. To perform activity recognition, different kinds of sensors have to be deployed in human-populated environments to monitor inhabitants' behaviors and capture environmental changes generated by human actions. The information provided by those sensors has to be processed through data analysis techniques and/or knowledge representation formalisms to create appropriate activity models and subsequently use them for activity recognition.

The literature shows two main approaches to solve activity recognition, namely the data-driven and knowledge-driven approaches. Data-driven approaches use large-scale datasets of sensors to learn activity models using data mining and machine learning techniques, whereas knowledge-driven approaches exploit rich prior knowledge in the domain of interest to build activity models using knowledge engineering and management technologies.

For knowledge-driven activity recognition systems, a widely recognized drawback is that activity models are usually static, i.e. once they are defined, they cannot be automatically adapted to users' specificities [5]. This is a very restrictive limitation, because in general it is not possible to define complete activity models for every user. Domain experts have the necessary knowledge about activities, but this knowledge may not be enough to generate complete models. To make knowledge-driven activity recognition systems work in real world applications, activity models have to evolve automatically to adapt to users' varying behaviors. It turns out that model adaptability and evolution are aspects that can be properly addressed by data-driven approaches. Hence, the objective of this paper is to use data-driven techniques to **make knowledge-driven activity models evolve automatically** based on sensor data generated by specific users.

Let us illustrate it with an example. Figure 1 shows an initial activity model for the *MakeCoffee* activity, which is composed by the actions *hasCoffee* and *hasContainer*. These are the necessary actions for every person to perform a *MakeCoffee* activity, which highlight the indispensable actions of making a coffee, i.e. the use of coffee and a container. However, some users might add some milk and sugar, others cream etc. The idea of our approach is to create high-level activity models with only these indispensable actions, and then use the data generated by a specific user performing the

Figure 1: Illustrative example of the objective of the paper: using initial model for *MakeCoffee* and user generated data, the learning algorithm learns 2 specialized and complete models.

activity to learn those new actions. In the case of Figure 1, where the initial activity model will only include coffee and container, the system would learn that *MakeCoffee* is performed in two ways by the user: in the first one, the user adds milk (*hasMilk*) and sugar (*hasFlavor*), while in the second one only sugar is added (*hasFlavor*). Hence two **specialized and complete activity models** of *MakeCoffee* can be learned. This way, initially experts only have to provide incomplete activity models with necessary actions. Afterwards the learning system can analyze a user’s behavioral data and learn the specialized and complete models to enrich the knowledge-base, thus improving initial activity models.

The scientific contributions presented in this paper are:

1. An **activity clustering algorithm** which uses initial activity models and context knowledge to recognize action clusters that form an activity and aggregate new actions.
2. A **learning algorithm** that uses activity clusters to learn specialized and complete activity models for every defined activity.

The paper is structured as follows: Section 2 presents the related work. Section 3 introduces the basic theory of ontology-based activity modeling. Section 4 describes in detail the proposed approach to learn specialized and complete activity models, followed by the evaluation and results obtained in

Section 5. Section 6 discusses the results and the paper concludes in Section 7 with conclusions and future work.

2. Related Work

Sensor-based activity recognition is based on the use of emerging sensor network technologies for activity monitoring. The generated sensor data from sensor-based monitoring are mainly time series of state changes and/or various parameter values that are usually processed through data fusion, probabilistic or statistical analysis methods and formal knowledge technologies for activity recognition. There are two main approaches for sensor-based activity recognition in the literature: data-driven and knowledge-driven approaches. An exhaustive review can be found in [6].

The idea behind data-driven approaches is to use data mining and machine learning techniques to learn activity models. It is usually presented as a supervised learning approach, where different techniques have been used to learn activities from collected sensor data. Data-driven approaches need big data bases of labeled activities to train different kinds of classifiers. The learning techniques used in the literature are broad, going from simple Naive Bayes classifiers as in [7] to Hidden Markov Models [8], Dynamic Bayesian Networks [9], [10], and Support Vector Machines [11].

The need of large-scale data bases has led to the *cold-start* problem for the data-driven activity recognition approach. Rashidi and Cook tried to overcome the problem of depending on manually labeled activity data bases in [12]. They use a non-labeled data base, where they extract activity clusters using unsupervised learning techniques. Those clusters are used to train a boosted HMM, which is shown to be able to recognize several activities. However, the approach still suffers from some other typical problems of data-driven approaches such as activity model generalizing.

Knowledge-driven activity modeling is based on real world observations that the list of objects and functionalities to perform an activity are always very similar. For example, to prepare coffee, a liquid container is needed alongside with some coffee and sugar. Even though different people may use different coffee brands, some may add milk and some may prefer brown sugar to white sugar, there are some essential concepts that are always present for every activity. The idea is to use this prior knowledge to create rough activity models. The implicit relationships between activities, related temporal

and spatial context and the entities involved (objects and people) provide a diversity of hint and heuristics for inferring activities.

The first step for knowledge-driven systems is to acquire the needed contextual knowledge. This is usually achieved using standard knowledge engineering approaches. Afterwards, knowledge structures will be computationally modeled using a formal knowledge representation formalism, e.g. schemas, rules or networks. Depending on the nature of the acquired knowledge, different approaches can be distinguished. Some researchers use logic-based approaches for activity recognition, as Bouchard et al. [13] and Chen et al. [14]. Others adopt ontology-based approaches which allow a commonly agreed explicit representation of activity definitions independent of algorithmic choices, thus facilitating portability, interoperability and reusability. Good examples can be found by Riboni et al. in [15], [16] and by Chen et al. in [17].

Knowledge-driven approaches have a number of advantages, including (i) the avoidance of the *cold-start* problem by means of modeling activities from knowledge acquired by experts rather than from data, (ii) the generality and applicability of the models, since activities are defined by their intrinsic features and not by how certain people perform them and (iii) making systems that are semantically clear and understandable for human beings. However, they also have some disadvantages: (i) they are generally weak in handling temporal aspects of activities and sensor uncertainty, and (ii) activity models are static, i.e. they do not evolve in time and hence they are not adaptable.

Chen et al. present in [5] an ontology-based hybrid approach to activity modeling, where learning techniques are developed to learn specific user profiles. However, their approach is limited to learn *descriptive properties* such as the time and duration an activity is performed or the concrete order of its constituent actions. They cannot learn action properties, so they assume that activity models in terms of actions are complete.

The work presented in this paper aims at overcoming the problem of learning new actions for activity models, since providing complete activity models for every user is generally impossible.

3. Ontology-Based Activity Modeling

The approach presented in this paper is based on the **dense-sensing paradigm**, which establishes the idea of inferring activities by monitoring

human-object interactions through the usage of multiple multi-modal miniaturized sensors. Single-user single-activity scenarios are considered, where only one user is monitored and concurrent activities cannot be performed.

In this scenario, ontology-based activity recognition systems have shown to perform robustly [17]. Central to those approaches is the ontology-based activity modeling. Activities are defined as ontological concepts and all actions that are required to perform the activity as the properties of the concept. For example, making tea involves taking a cup from the cupboard, putting a teabag into the cup, adding hot water to the cup, then milk and/or sugar. The ontological model of making tea, i.e. *MakeTea* concept, can be defined by action properties *hasContainer*, *hasTeabag*, *hasHotwater*, *hasMilk* and *hasFlavor* in conjunction with descriptive properties such as activity start time *actStartTime* and duration *actDuration*.

Activities can be modeled at different levels of abstraction. As such, ontological activity concepts are usually organized in a hierarchical structure to form super-class and sub-class relationships. For example, *MakeTea*, *MakeCoffee* and *MakeHotChocolate* activities can be modeled as the subclasses of *MakeHotDrink* activity, which is in turn the subclass of *MakeDrink*. The main problem of this modeling approach is that obtaining complete models for every person is not generally possible.

Assuming those constraints, let us introduce some important definitions:

Sensor activation: a sensor activation occurs when a sensor changes its state from the no-interaction state to interaction state. Reverse transitions are not considered. For example, when a user takes a glass, the activation is tagged as *glassSens*.

Type: when referring to activities and objects, type captures a purpose based classification. Considered classes are *Cooking*, *Hygiene*, *Entertainment* and *Housework*. An activity can only have one type, while objects can have more than one. When referring to sensors, type classifies sensors between *contact*, *electric*, *pressure* and *tilt* sensors.

Actions: actions are the primitives of activities and are directly linked to sensor activations. For example, *cupSens* and *glassSens* are linked to the action *hasContainer*. A sensor activation can only be mapped to one single action.

Initial Activity Model (IAM): activity models are sequences of actions defined by a domain expert. Initial activity models refer to the minimum number of necessary actions to perform an activity. The objective of such models is to represent incomplete but generic activity models. Initial activity

models also have an estimation of the maximum duration of the activity.

Extended Activity Model (EAM): a complete and specialized version of an IAM. By *complete* we mean that an activity model contains all the actions performed by a user for the corresponding activity. By *specialized* we mean there are two or more different complete action sequences for the corresponding activity, i.e. specialized sub-classes of that activity exist.

4. Approach to Learning Extended Activity Models

Based on the shown constraints and definitions, an offline system for EAM learning is presented. The inputs of the system are:

1. Context knowledge: provided by a domain expert. It contains prior knowledge about activities (type, location and IAM), objects of the environment (type, location and attached sensors) and sensors (type, described action and object to which it is attached).
2. Sensor activation dataset: an unlabeled time-stamped sensor activation dataset with the activity traces of a concrete user.

The output of the presented approach is a list of action sequences for each activity, which describe the EAMs for each activity.

4.1. Detailed Design

The designed system architecture to learn EAMs is depicted in Figure 2. Firstly, a sensor activation dataset has to be collected. A real smart environment or a synthetic data generator can be used to get this dataset. Afterwards, a novel clustering process is run, using the context knowledge provided by an expert. The clustering process is divided into two steps: (i) the Semantic Activity Annotation algorithm (SA^3) uses IAMs to detect activities in the unlabeled dataset and initialize activity clusters, and (ii) the Activity Clustering algorithm (AC) uses activity, object and action knowledge to expand initial activity clusters detected by SA^3 . As a result, several action clusters for every activity are generated. Those clusters are finally processed by Activity Model Learner (AML), which filters incorrect action sequences and outliers to learn final EAMs as a list of action sequences.

Figure 2: The detailed design architecture of the proposed approach. All modules are implemented in Python 2.7.

4.2. Semantic Activity Annotation (SA^3)

The SA^3 algorithm is used as the initialization step of a clustering process. As explained in [18], it uses IAMs to run a specially designed pattern recognition algorithm, where IAMs act as the patterns to be recognized. It works with positive sensor noise (sensors that are activated when they should not), missing sensor noise (sensors that are not activated when they should) and varied order of actions.

A new feature has been added to SA^3 : **activity location inference**. Let $S = \{a, b, c, d\}$ be a sequence of actions, where a and d pertain to the IAM of activity A . The sequence S is an activity A with location L iff a and d have been executed in the same location and that location is compatible with A activity’s locations. As an example, imagine actions $hasBook(book-a)$ and $useFurniture(sofa)$ pertain to the IAM of *ReadBook*. *ReadBook* can be performed in the lounge or in the bedroom, according to context knowledge, but not in both. As *book-a* object is in the bedroom but *sofa* object is in the lounge, the sequence containing $hasBook(book-a)$ and $useFurniture(sofa)$ cannot form an activity. Activity location inference has shown to decrease SA^3 algorithm’s false positive rates considerably.

The result of SA^3 is a sensor activation dataset where actions describing an activity are labeled with that activity name. All the other actions will be labeled with the special label *None*.

4.3. Activity Clustering (AC)

Initial clusters provided by SA^3 are expanded in this step, using context knowledge and time-based metrics for action aggregation. Assume Figure 3 shows the output of SA^3 for a concrete sequence of actions. Dashes are time intervals without any action. Circles represent actions that are in one

or more IAMs (two circles do not necessarily have to be the same action). Crosses are actions that are not included in any IAM.

Figure 3: Output of SA^3 for a concrete sequence of actions.

SA^3 detects activities A_1 and A_2 in that sequence of actions. That output has to be interpreted as the initialization of a clustering algorithm, so only actions that are in the IAM of the detected activity can be really considered part of that activity. Every action that is inside A_1 or A_2 but is not in their IAMs is considered an **insider**, while every action out of detected activities is an **outsider**.

Due to single-user single-activity scenario constraint, an insider may pertain to its wrapping activity or to none, i.e. it has been produced by noise. But an outsider can pertain to its previous activity, next activity or to none. This fact demands a different treatment for both cases. AC first treats all insider actions and afterwards computes all outsiders using different approaches.

Insiders: to decide whether an insider has to be added to its wrapping activity, a compatibility function between an activity and an action is defined:

$$Comp(A, a) = Loc(A, a) \wedge Type(A, a) \quad (1)$$

where A is an activity and a is an action. $Loc(A, a)$ is the location compatibility between an activity and an action, defined as in Section 4.2. $Type(A, a)$ is the type compatibility between an activity and an action. It is calculated as the intersection between the list of types of the object which has been mapped to action a and the type of the activity A . Type information for objects and activities is in the context knowledge.

Hence an insider action a will only be aggregated to its wrapping activity A , if $Comp(A, a) = True$, i.e. the insider has been executed in the same location of the activity, and its purpose is compatible with the activity type.

Outsiders: as an outsider can be aggregated to its previous or next activity, first of all the algorithm checks the feasibility of both options, defining the *candidate function*:

$$Cand(A, a) = Comp(A, a) \wedge InRange(A, a) \quad (2)$$

An activity A is a candidate activity for action a , if they are compatible ($Comp(A, a) = True$) and in range ($InRange(A, a) = True$). *InRange* function captures the time feasibility. For example, if an action has been executed two hours before an activity whose estimated duration is three minutes, it should not be aggregated to that activity. To capture time feasibility, the duration given by the expert in an IAM is interpreted as the standard deviation for concrete executions of that activity, i.e. the vast majority of the activities executed by any user, will lie inside the time area limited by the duration. This can be seen in Figure 4. So time distances among actions pertaining to a concrete activity are modeled by a Gaussian distribution, where the duration estimation given by the expert for that activity is the standard deviation. A Gaussian distribution has been selected, because it captures perfectly the idea of activity duration as a time estimation where the majority of executions of that activity occur. The probability for actions of an activity to lie in the area limited by the duration estimation is very high and gets lower as it gets further from that duration.

Nevertheless, due to human behavior variations, there will be some executions whose duration is outside the standard deviation. To capture those executions, *InRange* function considers all the actions lying inside two standard deviations. For a concrete activity execution, the mean of the activity is calculated as the center of the detected start and end of the activity, as given by SA^3 . Hence, in the case depicted in Figure 4, the outsider action is in range with activities A_1 and A_2 .

Using the candidate function, the following cases can be faced for an outsider action a and surrounding activities A_1 and A_2 :

1. $Cand(A_1, a) = Cand(A_2, a) = False \rightarrow a$ is noise
2. $Cand(A_1, a) = True \wedge Cand(A_2, a) = False \rightarrow$ aggregate a to A_1
3. $Cand(A_1, a) = False \wedge Cand(A_2, a) = True \rightarrow$ aggregate a to A_2
4. $Cand(A_1, a) = Cand(A_2, a) = True \rightarrow$ need of a new heuristic

For the fourth case, a new heuristic is defined that states that an outsider action a will be aggregated to the time closest activity:

$$\min\{\Delta_t(A_1, a), \Delta_t(A_2, a)\} \quad (3)$$

Figure 4: Gaussian distributions of activity durations for activities A_1 and A_2 . Vertical lines show the standard deviation and the star represents an outsider action.

To implement this heuristic, a definition for $\Delta_t(A, a)$, the time distance between an activity and an action, is needed. As an activity has a duration and an action is described by a time instant, three time metrics are proposed:

1. Simple time distance: the distance between the center of the activity as given by SA^3 and the action time coordinate.

$$\Delta_t(A, a) = |C_A - t_a|, \text{ where } C_A = \frac{t_{end}^{SA^3} - t_{start}^{SA^3}}{2} \quad (4)$$

2. Normalized time distance: simple time distance normalized by the duration of the activity as given by the expert.

$$\Delta_t(A, a) = \frac{|C_A - t_a|}{Duration_A}, \text{ where } C_A = \frac{t_{end}^{SA^3} - t_{start}^{SA^3}}{2} \quad (5)$$

3. Dynamic center normalized time distance: only used for previous activity, it dynamically calculates the center of the activity depending on already aggregated actions.

$$\Delta_t(A, a) = \frac{|C_A - t_a|}{Duration_A}, \text{ where } C_A = \frac{t_{start}^{AC} + Duration_A}{2} \quad (6)$$

Let us explain the third time distance. As outsiders are treated in time order for convenience, when treating outsider a , its previous activity's previous actions have already been treated. This means that the start of that previous activity has been fixed. In contrast with time metrics 1 and 2, where

activity start and end were given by SA^3 and activity duration was assumed to be located symmetrically around the center of the activity, the third time metrics uses the start time of the previous activity as found by AC . Afterwards, the center of the activity is calculated projecting the duration from the starting point. This makes previous activity treatment more accurate. However, notice that the same cannot be applied to the next activity, since it has not been yet treated. Hence, the best guess is to keep using start and end times provided by SA^3 .

To sum up, the AC algorithm takes the results of SA^3 . First, it treats insiders for all the activities detected by SA^3 , using the compatibility function. Afterwards, it treats outsiders in time order, using the candidate function and defined three time metrics. The output of the algorithm is a labeled sensor activation dataset and a file where all activity clusters and their occurrence frequency can be found.

4.4. Activity Model Learner (AML)

Activity clusters extracted by the clustering process contain all the action sequences performed by a user for each activity. But some of those clusters are spurious, due to sensor noise, user erratic behavior and clustering errors. The objective of AML is to remove spurious action sequences. For that purpose, a two-step algorithm has been designed and implemented. For all the clusters extracted for an activity, the following steps are performed:

1. Fuse equal action sequences: some action sequences contain the same actions, but in different orders. For example, $S_1 = \{a, b, c\}$ and $S_2 = \{b, c, a\}$. The order of actions is not important for activity models, so both sequences are fused. To detect equal sequences, the Jaccard coefficient is used [19]. The Jaccard coefficient between two sequences A and B is defined as:

$$Jaccard(A, B) = \frac{A \cap B}{A \cup B} \quad (7)$$

Any two sequences whose Jaccard coefficient is 1 are fused. Fusing means removing one of the sequences and adding frequencies to store in the remaining sequence.

2. Run Jaccard based outlier detection algorithm, which has been specially designed to learn proper action sequences.

The Jaccard based outlier detection is an iterative algorithm. It calculates the so called *Jaccard Matrix* (JM), which is a square matrix of remaining action sequences. $JM_{i,j}$ stores the Jaccard coefficient for action sequences i and j . The diagonal of JM is 1, since two equal action sequences' Jaccard coefficient is 1. Removing diagonal values $\hat{J}M$ is obtained, which is used to calculate the median and the standard deviation to the median. The median is used rather than the mean value, since the median is robust to outliers. Using those statistics a threshold θ is calculated, such that:

$$\theta = \max\{\text{median}(\hat{J}M) + \text{std}(\hat{J}M), \lambda\} \quad (8)$$

The first part of the calculation of θ captures the relative similarity among all action sequences and establishes an adequate threshold to identify outliers. However, as the Jaccard coefficient is defined in an absolute scale, the second part ($\lambda \in [0, 1]$) has to be added. For relatively short action sequences used in the experiments (the longest ones are around 9 actions), 0.75 has shown to be a good balanced value. λ prevents fusing sequences that even being more similar than most of the others, their similarity is not higher than it. Sequences below λ are considered too different to be fused.

The algorithm fuses sequences whose Jaccard coefficient is higher than θ , until no sequences can be fused. To fuse, the sequence with lowest frequency is removed and its frequency value is added to the frequency of the other sequence. This fusing heuristic states that the lower frequency sequence is a spurious variation of the higher frequency sequence.

5. Evaluation and Results

5.1. Evaluation Methodology

To evaluate the presented approach activity datasets that fulfill several conditions are needed: (i) contain a lot of samples of different activities, (ii) sensor activations must be labeled in order to provide a solid ground truth, (iii) at least some of the activities have to be performed in varied ways, and (iv) several users are needed as the approach aims at capturing concrete activity models for concrete users.

Datasets that meet all those conditions cannot be found in public repositories. Furthermore, setting up experiments with real users to cover all the needs enumerated is very expensive and time consuming. To overcome those problems a novel experimental methodology has been designed. First of all a

survey has been circulated among several people to capture how they perform certain activities. For that purpose, Google Forms has been selected, because surveys can be sent by e-mail and answers are anonymous. The survey used for the experiments of this paper can be found in the web¹. Based on their answers, individual scripts have been prepared for a synthetic dataset generator tool, described in [18], where sensor activation patterns for each activity can be specified. This experimental methodology allows generating easily as many datasets as needed, where all sensor activations are properly labeled and different users performing activities in several ways are available.

Using this methodology, two evaluation scenarios have been set: the **ideal** and the **complete** scenario. The first one does not contain any sensor noise and time lapses between contiguous activities are greater than expected duration, which makes easier the learning process. The complete scenario is closer to reality. It has sensor noise and activity time lapses are realistically set, which makes learning more demanding.

The synthetic dataset generator tool provides two ways to model sensor noise: (i) positive sensor noise, where a sensor that was not supposed to activate gets activated, and (ii) missing sensor noise, where a sensor that should have been activated fails. The first noise model is used to model sensor failures and user’s erratic behavior, i.e. a user that interacts with objects even though they are not used for the ongoing activity. Sensor failure models are obtained from [17], where authors provide a detailed study of sensor failure statistics. Noise models have been specified depending on sensor type.

So the experimental set-up consists on: (i) 8 real users, (ii) 7 activities of daily living labeled as *MakeCoffee*, *MakeChocolate*, *MakePasta*, *BrushTeeth*, *WatchTelevision*, *WashHands* and *ReadBook* and (iii) 2 scenarios, the ideal and the complete. Results have been evaluated in two ways: (i) compare the labels given by the clustering process to every sensor activation with the ground truth produced by the synthetic dataset generator tool by means of true positives, false positives and false negatives, and (ii) compare learned activity models with the models provided by users in their answers to the survey using again the same metrics.

¹<http://goo.gl/etCNyi>

5.2. Results

All the experiments run produce datasets of 60 days per user, both for the ideal scenario and the complete scenario. A typical user dataset for the ideal scenario contains around 2400 sensor activations, while the complete scenario has around 3500. Datasets used for the experiments are available in the web². The same context knowledge is used for all users and experiments, hence IAMs are identical. As differences among 3 time metrics are not very big, results only show the best approach: use dynamic center normalized time distance (equation 6) for previous activity and normalized time distance (equation 5) for next activity.

Tables 1 and 2 show the results of the clustering process, comparing the labels assigned by SA^3 and AC with the ground truth. The average results for all users are shown in each table. Standard deviation is quite significant for SA^3 , but it is very small for AC . That is why it is not shown in the tables. Table 1 shows the results for the ideal scenario and Table 2 for the complete scenario.

On the other hand, Tables 3 and 4 show the results of the complete learning process for EAMs. The output of the clustering process for the previous experiments are used to run the AML as explained in Section 4.4. Average results for all users are provided. While Table 3 shows the results for the ideal scenario, Table 4 depicts the results for the complete one.

6. Discussion

Tables 1 and 2 show the good performance of the activity clustering process using context knowledge. True positive rate is very high for all activities. The lowest rate is found for activity *ReadBook* in the complete scenario: 94.6%. This is due to two factors: (i) the missing sensor probability for pressure sensors is around 10% and (ii) the IAM for *ReadBook* has the action *useFurniture*, which is produced by pressure sensors. All other activities, even in the complete scenario, show true positive rates higher than 97%, reaching in some cases 100% rates. Those high rates are accompanied with very low rates of false positives and negatives. For example, the highest false positive rate has been found for activity *MakePasta* in the complete scenario with 5.6%.

²http://www.morelab.deusto.es/pub/synthetic_adl_datasets/

Activity	Clustering Results					
	True Positive (%)		False Positive (%)		False Negative (%)	
	SA^3	AC	SA^3	AC	SA^3	AC
MakeChocolate	55.05	98.67	0	0	44.95	1.33
WatchTelevision	75.12	100	0	0	24.88	0
BrushTeeth	90.91	98.27	0	0	9.1	1.73
WashHands	74.93	99.72	0.1	5.1	25.07	0.27
MakePasta	55.74	99.8	0	0.04	44.26	0.2
ReadBook	89.08	100	0	0	10.91	0
MakeCoffee	62.63	99.09	0	0	37.37	0.91

Table 1: Average results for 8 users of the clustering process for the **ideal scenario**.

Activity	Clustering Results					
	True Positive (%)		False Positive (%)		False Negative (%)	
	SA^3	AC	SA^3	AC	SA^3	AC
MakeChocolate	54.73	97.76	1.2	2.64	45.27	2.24
WatchTelevision	71.13	100	0	0	28.87	0
BrushTeeth	91.18	98.43	0.28	0.41	8.82	1.57
WashHands	75.12	98.37	0.69	4.55	24.88	1.63
MakePasta	53.88	99.39	1.34	5.62	46.12	0.61
ReadBook	82.91	94.66	0.45	0.3	17.09	5.34
MakeCoffee	59.14	99.24	1.32	2.6	40.86	0.76

Table 2: Average results for 8 users of the clustering process for the **complete scenario**.

Activity	Learning Results			Average Number of Patterns
	TP (%)	FP (%)	FN (%)	
MakeChocolate	100	0	0	1
WatchTelevision	100	0	0	1.14
BrushTeeth	100	37.5	0	1.25
WashHands	100	25	0	1
MakePasta	100	0	0	2
ReadBook	100	0	0	1.12
MakeCoffee	100	0	0	1.71

Table 3: Average results for 8 users of the EAM learning process for **the ideal scenario**.

Activity	Learning Results			Average Number of Patterns
	TP (%)	FP (%)	FN (%)	
MakeChocolate	100	120	0	1
WatchTelevision	100	78.57	0	1.14
BrushTeeth	100	93.75	0	1.25
WashHands	100	75	0	1
MakePasta	100	56.25	0	2
ReadBook	100	12.5	0	1.12
MakeCoffee	100	100	0	1.71

Table 4: Average results for 8 users of the EAM learning process for **the complete scenario**.

It is worth to point out the behavior of the clustering process, divided into two steps. Tables 1 and 2 show how SA^3 labels correctly varying number of sensor activations. This number depends on the relation between the number of sensor activations performed by a user and the number of actions in the IAMs. It can be seen that for low action number activities like *Brush-Teeth* and *ReadBook*, SA^3 shows quite a high true positive rate and low false negative rate. However, activities like *MakeChocolate* or *MakePasta* have a true positive rate below 60% and high false negative rates. In any case, in the second step run by AC , true positives rise, false negatives get very low and false positives slightly increase. This means that SA^3 discovers activities' time locations very accurately and AC treats insider and outsider actions properly to achieve very good rates building on the results of SA^3 .

As far as learning EAMs concerns, the first fact shown by Tables 3 and 4 is that true positives for all activities in both scenarios are 100%. This means that the learning algorithm learns properly all the activity patterns performed by any user even in noisy scenarios. However, specially for the complete scenario, this result comes with high rates of false positives. Those rates were expected, since the objective of the learning algorithm is to avoid removing any activity pattern that has been actually performed by the user. It is preferable to get false positive patterns than removing any activity pattern that has been really performed. For that purpose, the learning process is conservative when removing and fusing activity patterns. Even with this conservative approach, it has been observed during experiments that the learning process can reduce clusters provided by AC from 17 to 3 in some cases, thus removing many false positives and keeping true positive rates of 100%.

Nevertheless, false positive rates have to be properly interpreted. The highest rate can be found for activity *MakeChocolate* in the complete scenario: 120%. As shown in the table, the average number of patterns of that activity is 1, which means that all 8 users perform it in only one way. So the EAM learning algorithm learns additional 1.2 patterns in average for the activity *MakeChocolate*. Putting it in that perspective, false positive rates are assumable.

Regarding false positive activity patterns, it has to be said that they are usually easy to discard for an expert for two reasons: (i) they have very low occurrence frequencies and (ii) they usually contain actions that are not generally executed for those activities. For example, for *MakeChocolate*, actions like *hasBacon* have been seen. Those actions cannot be discarded by AC ,

since they are type and location compatible with *MakeChocolate* (please, notice the slight false positive increments from ideal scenario to complete scenario in Tables 1 and 2 due to noise). Adding more knowledge to the context knowledge would allow discarding such actions in *AC*. But this brings the **initial knowledge balance problem**: how much knowledge should be initially provided to such a learning system? The answer depends a lot on the domain. The approach presented in this paper follows the philosophy of minimizing initial knowledge as much as possible. We believe results shown in Section 5.2 support this decision.

7. Conclusions and Future Work

This paper has introduced a novel approach to enhancing the completeness and accuracy of knowledge-driven activity models through incremental data-driven learning. The approach makes possible using incomplete activity models in a knowledge-driven activity recognition system to learn specialized and complete activity models. Central to the approach is the two-step clustering process, which has the singularity of using previous knowledge in order to find activity clusters and label them with the appropriate activity label. Those clusters are then treated by the learning algorithm in order to acquire extended activity models for different users.

Initial results as shown in Section 5.2 have suggested that the approach works well in realistic experimental set-ups, with real users' inputs, demanding time intervals for activity sequences and sensor noise. It has to be stressed that all the varying ways of performing an activity by a user are correctly captured by the approach with a success rate of 100% for all users and scenarios. Specialized and complete models for initial incomplete activity models can be properly learned automatically with minimum previous context knowledge.

It has been noticed that positive sensor noise has more pernicious effects on learned patterns than missing sensor noise. The latter makes clustering process true positive rates decrease, but it does not affect the learned activity patterns, whereas the former generates spurious activity patterns which cannot always be discarded by the learner. As positive noise is usually due to user erratic behavior, future research will try to address sensor processing steps to distinguish between real object usage and meaningless object interactions.

Another promising future research direction is to extend the learning approach to single-user concurrent-activities scenario. People do not usually

perform activities sequentially, but they tend to interleave activities, such as washing dishes while preparing pasta. This will have an impact in the clustering process, demanding more complex pattern recognition and time management.

8. References

- [1] T. Choudhury, S. Consolvo, The mobile sensing platform: An embedded activity recognition system, *IEEE Pervasive Computing* 7 (2) (2008) 32–41.
- [2] T. Fong, I. Nourbakhsh, K. Dautenhahn, A survey of socially interactive robots, *Robotics and autonomous systems* 42 (3) (2003) 143–166.
- [3] D. Weinland, R. Ronfard, E. Boyer, A survey of vision-based methods for action representation, segmentation and recognition, *Computer Vision and Image Understanding* 115 (2) (2011) 224–241.
- [4] K. V. Laerhoven, K. Aidoo, Teaching context to applications, *Personal and Ubiquitous Computing* 5 (1) (2001) 46–49.
- [5] L. Chen, C. Nugent, G. Okeyo, An Ontology-based Hybrid Approach to Activity Modeling for Smart Homes, *IEEE Transaction on Human-Machine Systems* 44 (1) (2014) 92–105.
- [6] L. Chen, J. Hoey, C. Nugent, D. Cook, Z. Yu, Sensor-based activity recognition, *IEEE Transactions on Systems, Man, and Cybernetics-Part C* 42 (6) (2012) 790–808.
- [7] L. Bao, S. Intille, Activity recognition from user-annotated acceleration data, in: L. N. i. C. Science (Ed.), *Pervasive Computing*, Springer Berlin Heidelberg, Vienna, Austria, 2004, pp. 1–17.
- [8] A. Galata, N. Johnson, D. Hogg, Learning structured behaviour models using variable length Markov models, in: *IEEE International Workshop on Modelling People*, IEEE, 1999, pp. 95–102.
- [9] M. Brand, N. Oliver, A. Pentland, Coupled hidden Markov models for complex action recognition, in: *Proceedings of Computer Vision and Pattern Recognition*, IEEE, 1997, pp. 994–999.

- [10] N. Oliver, A. Garg, E. Horvitz, Layered representations for learning and inferring office activity from multiple sensory channels, *Computer Vision and Image Understanding* 96 (2) (2004) 163–180.
- [11] O. Brdiczka, Learning situation models in a smart home, *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 39 (1) (2009) 56–63.
- [12] P. Rashidi, D. Cook, Discovering activities to recognize and track in a smart environment, *Knowledge and Data Engineering, IEEE Transactions on* 23 (4) (2011) 527–539.
- [13] B. Bouchard, S. Giroux, A. Bouzouane, A Smart Home Agent for Plan Recognition of Cognitively-impaired Patients., *Journal of Computers* 1 (5) (2006) 53–62.
- [14] L. Chen, C. Nugent, A logical framework for behaviour reasoning and assistance in a smart home, *International Journal of Assistive Robotics and Mechatronics* 9 (4) (2008) 20–34.
- [15] D. Riboni, L. Pareschi, Is ontology-based activity recognition really effective?, in: *Pervasive Computing and Communications Workshops (PERCOM Workshops), IEEE International Conference on*, 2011.
- [16] D. Riboni, C. Bettini, COSAR: hybrid reasoning for context-aware activity recognition, *Personal and Ubiquitous Computing* 15 (3) (2011) 271–289.
- [17] L. Chen, C. Nugent, H. Wang, A knowledge-driven approach to activity recognition in smart homes, *Knowledge and Data Engineering, IEEE Transactions on* 24 (6) (2012) 961–974.
- [18] G. Azkune, A. Almeida, D. López-de Ipiña, L. Chen, A Knowledge-Driven Tool for Automatic Activity Dataset Annotation, in: *IEEE Intelligent Systems, Warsaw, 2014*, p. In Press.
- [19] A. Jain, R. Dubes, *Algorithms for clustering data*, Prentice Hall, 1988.